PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	Attorney Docket No: SUNMP016
KINZHALIN et al.	Examiner: Zhen, Wei Y.
Application No: 09/881,791	Group Art Unit: 2191
Filed: June 14, 2001	Date: January 17, 2006
For: System and Method for Automated Assertion Acquisition in a Java Compatibility Testing Environment	

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on **January 17, 2006.**

Signed: _____
Kenneth D. Wright

TRANSMITTAL OF APPEAL BRIEF
(PATENT APPLICATION -- 37 CFR 192)

Mail Stop: Appeal Brief-Patents
Commissioner for Patents
Alexandria, VA 22313-1450

Sir:

This Appeal Brief is in furtherance of the Notice of Appeal filed in this case on October 17, 2005.

This application is on behalf of:

☐ Small Entity ☒ Large Entity

Pursuant to 37 CFR 1.17(f), the fee for filing the Appeal Brief is:

☐ $250.00 (Small Entity) ☒ $500.00 (Large Entity)

☐ Appeal Brief Fee has already been paid. Prosecution was re-opened by Examiner in response to the Appeal Brief, filed _____.

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136 apply:

☒ Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d)) for the total number of months checked below:

01/23/2006 WABDELR1 00000042 09881791
01 FC:1402 500.00 OP
02 FC:1251 120.00 OP

Months	Large Entity	Small Entity
☒ one	$120.00	$60.00
☐ two	$450.00	$225.00
☐ three	$1,020.00	$510.00
☐ four	$1,590.00	$795.00

If an additional extension of time is required, please consider this a petition therefor.

☐ An extension for __ months has already been secured and the fee paid therefore of $ is deducted from the total fee due for the total months of extension now requested.

☐ Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that Applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Total Fees Due:

Appeal Brief Fee	$500.00
Extension Fee (if any)	$120.00
Total Fee Due	**$620.00**

☒ Enclosed is Check No. 15652 in the amount of $620.00.

☐ The Commissioner is authorized to charge the total fees due of $___ to Deposit Account No. 50-0850, (Order No. _____).

☒ The Commissioner is authorized to charge any additional required fees or credit any overpayment to Deposit Account No. 50-0850, (Order No. SUNMP016).

One additional copy of this transmittal is enclosed for fee processing.

Respectfully submitted,
MARTINE PENILLA & GENCARELLA, LLP

Kenneth D. Wright
Reg. No. 53,795

710 Lakeway Drive, Suite 200
Sunnyvale, CA 94085
(408) 749-6900
Customer No. 32,291

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

EX PARTE KINZHALIN et al.

Application for Patent

Filed June 14, 2001

Application No. 09/881,791

FOR:

System and Method for Automated Assertion Acquisition in a Java Compatibility Testing Environment

---

APPEAL BRIEF

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on January 17, 2006.

Signed: _____
Kenneth D. Wright

MARTINE PENILLA & GENCARELLA, LLP
Attorneys for Applicants

TABLE OF CONTENTS

Page No.

I. REAL PARTY IN INTEREST

The real party in interest is Sun Microsystems, Inc., the assignee of the present application.

II. RELATED APPEALS AND INTERFERENCES

The Applicants are not aware of any related appeals or interferences.

III. STATUS OF THE CLAIMS

A total of 20 claims were presented during prosecution of this application. The Applicants appeal rejected claims 1-20.

IV. STATUS OF THE AMENDMENTS

In response to the Final Office Action dated June 16, 2005, an Amendment was filed on August 16, 2005. In an Advisory Action dated September 15, 2005, the Examiner failed to indicate whether the Amendment of August 16, 2005, had been allowed after-final entry. Because the Examiner has not indicated otherwise, the Applicants assume that the Amendment of August 16, 2005, has not been allowed entry. In the Advisory Action of September 15, 2005, the Examiner indicated that claims 1-20 remain rejected.

V. SUMMARY OF THE INVENTION

The present invention provides methods for tracking a specification of a computer program to automatically obtain testable assertions from the specification of the computer program and validate the obtained testable assertions. (p. 6, lines 1-5) The embodiments of the present invention allow a test developer to perform testing routines in a semi-automated way that improves performance, reduces human error, and allows the test developer to spend more time on test development itself. (p. 7, lines 11-13) Moreover, the

embodiments of the present invention produce various reports on how a technology compatibility kit (TCK) covers the corresponding specification. (p. 7, lines 14-15) These reports are very useful for project management since they allow the test developer to analyze the TCK completeness and plan future TCK works. (p. 7, lines 15-17)

Independent claim 1 recites one embodiment of the present invention directed to a method for automated acquisition of assertions in a specification of a computer program. The method includes an operation for receiving a specification of a computer program as an input, wherein the specification includes a plurality of sentences describing the computer program. The method further includes an operation for obtaining a sentence from the plurality of sentences within the specification that describes the computer program. In another operation, a determination is made as to whether the obtained sentence is a testable assertion that describes the behavior of an application programming interface (API) that can be tested. If the obtained sentence is a testable assertion, the obtained sentence is marked as testable. The method further includes an operation for using the sentences marked as testable to determine whether a test suite for testing the computer program is adequate.

Independent claim 8 recites one embodiment of the present invention directed to a computer readable media including program instructions for automatically obtaining assertions from a specification of a computer program. The computer readable media includes a code segment that receives a specification of a computer program as an input. Another code segment is provided to identify a context within the specification of the computer program. The identified context is then parsed by another code segment to obtain sentences. The computer readable media further includes a code segment that determines whether the obtained sentences are testable assertions. In the present embodiment, each testable assertion is a sentence that describes behavior of an API that can be tested.

Additionally, the computer readable media includes a code segment that adds the obtained sentences identified as testable assertions to an assertion result set. The assertion result set can be used to facilitate testing of the specification of the computer program.

Independent claim 14 recites another embodiment of the present invention directed to a computer readable media including program instructions for automated acquisition of assertions in a specification of a computer program. The computer readable media includes a code segment that receives a specification of a computer program in a text format. The specification of the computer program includes a plurality of sentences. Another code segment is provided for obtaining a sentence from the plurality of sentences within the specification of the computer program. The computer readable media further includes a code segment that determines whether the obtained sentence is a testable assertion that describes behavior of an API that can be tested. Additionally, the computer readable media includes a code segment that marks the obtained sentence as testable when it is determined that the obtained sentence is a testable assertion.

Support for the features of the claims as summarized above can be found throughout the Detailed Description of the application. However, for the Board's convenience, a brief summary of supporting portions of the Detailed Description is provided below.

Figure AB-1 is an illustration showing a flowchart of a process 210a for obtaining assertions from a specification of a computer program (the "specification"), in accordance with one embodiment of the present invention. (p. 18, lines 7-8) In an operation 502, the process 210a receives the specification as an input. (p. 18, line 12) After receiving a request to process the specification, the process 210a determines whether or not the specification is available. (p. 18, lines 14-15) If the specification is not available, a problem is reported in an operation 504, and the process 210a is aborted in an operation

505. (p. 18, lines 15-17) However, if the specification is available, the process 210a continues with an operation 506 in which a next context is identified within the specification. (p. 18, lines 17-19)

The context within the specification is a set of circumstances related to each assertion within the specification. (p. 18, lines 19-20) Each assertion within the specification has an associated context. (p. 18, lines 20-21) In one embodiment of the present invention, the specification has a tree-like structure. (p. 18, lines 21-22) For example, the API specification tree has a top-level specification as a root, and package-level specifications as immediate children of the root, with class-level specifications as their children, and finally constructor/method/field specifications as leaves. (p. 18, line 22, through p. 19, line3) An assertion is uniquely identified by both the assertion text and its associated context. (p. 19, lines 3-4) The assertion context can be defined as the associated node of the specification tree. (p. 19, lines 4-5) For example, an assertion from a package-level specification would mean that the package-level is the assertion context. (p. 19, lines 5-6) It should be noted, however, that embodiments of the present invention can process specifications that do not strictly adhere to this particular multi-level structure, because the specification tree can be considered to contain only one node. (p. 19, lines 6-9)

Following the operation 506, a decision is made in an operation 508 as to whether a context is available within the specification. (p. 19, line 10) If no context is available, the process 210a is completed in an operation 514. (p. 19, line 10-11) Generally, when no further context is available the input application has been processed and a corresponding assertion result set completed. (p. 19, lines 11-13) If the context is available, the process 210a continues with operation 509 in which the context is filtered. (p. 19, lines 13-15) Once the context portion of the specification is filtered, an operation 510 is performed to

parse the context portion of the specification. (p. 19, lines 17-18) Specifically, the context portion of the specification is parsed to find assertions present therein. (p. 19, lines 18-19)

The embodiments of the present invention scan through the specification and split the entire text into logical statements. (p. 19, lines 20-21) In some embodiments, each logical statement is examined by type to indicate if it is a testable assertion. (p. 19, lines 21-22) Statements are considered testable assertions if they are intended to describe behavior of an API that can be tested by the TCK. (p. 20, lines 1-2) Also, examples or sample code pieces that are provided in the specification are typically testable and can be verified by the TCK. (p. 20, lines 2-4) In this sense, examples or sample code are generally considered testable assertions. (p. 20, line 4) Further, it should be noted that some assertions can be implied or indirectly stated in the specification, and these should be identified as well for testing. (p. 20, lines 5-7)

On the other hand, statements intended to describe the behavior of an API, but which cannot be tested by the TCK due to the special nature of the behavior or functionality, are generally considered non-testable assertions. (p. 20, lines 8-10) Similarly, some statements will form general descriptions of the API such as a description of a package, class, method, or field, and so forth. (p. 20, lines 10-12) If such a general description does not describe behavior, but is aimed rather at providing a context for the rest of the text, then such a statement is not intended to be an assertion and should not be tested. (p. 20, lines 12-14) Hence, these statements are generally not considered to be assertions, as they are easy to misinterpret. (p. 20, lines 14-15) An exemplary context portion of the specification is shown in Table 1. (p. 20, line 16) A list of assertions based on the context specification shown in Table 1 is presented in Table 2. (p. 21, lines 11-12) Thus, Tables 1 and 2 illustrate one example of how an embodiment of the present

invention can parse a context portion of a specification and create a list of assertions based on that context. (p. 21, lines 27-29)

Having parsed the context specification, the discovered assertions are added to an assertion result set, in operation 512. (p. 21, lines 30-31) The process 210a then continues with another identify context operation 506. (p. 21, lines 31-32) In this manner, the process 210a can parse through an input specification and generate a list of assertions based on the input specification. (p. 21, lines 32-33) As mentioned above, in some embodiments of the present invention, each logical statement is examined by type to indicate if it is a testable assertion. (p. 21, lines 33-35)

It should be appreciated that the above discussion represents only a summary of the present invention. A more in-depth discussion of the present invention is provided in the Detailed Description section of the application.

Table 1

```
public static String toString(int i, int radix)
```

Creates a string representation of the first argument in the radix specified by the second argument.

If the radix is smaller than Character.MIN_RADIX or larger than Character.MAX_RADIX, then the radix 10 is used instead.

If the first argument is negative, the first element of the result is the ASCII minus character '-' ('\u002d').

If the first argument is not negative, no sign character appears in the result.

Parameters:

i - an integer.

radix - the radix.

Returns:

a string representation of the argument in the specified radix.

See Also:

Character.MAX_RADIX, Character.MIN_RADIX

Table 2

A1.	Creates a string representation of the first argument in the radix specified by the second argument.
A2.	If the radix is smaller than Character.MIN_RADIX or larger than Character.MAX_RADIX, then the radix 10 is used instead.
A3.	If the first argument is negative, the first element of the result is the ASCII minus character '-' ('\u002d').
A4.	If the first argument is not negative, no sign character appears in the result.

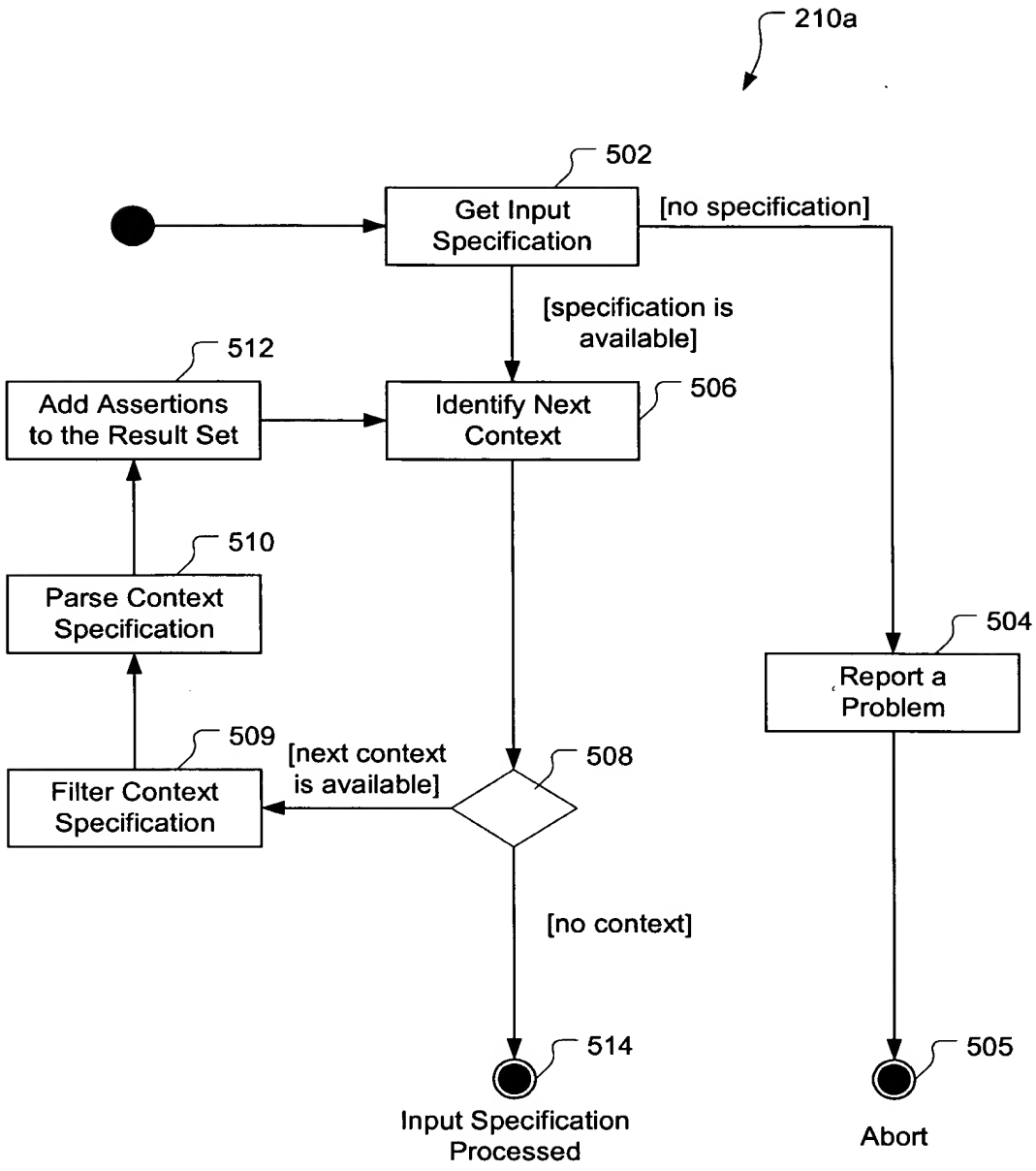

Fig. AB-1

(Fig. 5A of application)

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 8-20 stand rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement.

Claims 1-20 stand rejected under 35 U.S.C. 101 as being directed to non-statutory subject matter.

Claims 1-20 were rejected under 35 U.S.C. §103(a) as being unpatentable over Pavela (U.S. Patent No. 6,332,211) in view of Microsoft Press Computer Dictionary (MPCD).

VII. ARGUMENTS

A. Rejections of Claims 8-20 under 35 U.S.C. 112

The Examiner has asserted that the term "computer readable media" was not sufficiently described in the specification as originally filed. The present application is cross-referenced to U.S. Provisional Application No. 60/291,670. The present application also incorporates by reference the entire specification of U.S. Provisional Application No. 60/291,670. In the Amendment dated August 16, 2005, the second full paragraph on page 2 of U.S. Provisional Application No. 60/291,670 was incorporated within the present application immediately before the paragraph beginning at page 25, line 16. Because the incorporated paragraph was the only amendment made in the Amendment of August 16, 2005, the Applicants submit that the Amendment of August 16, 2005, should be allowed after-final entry to place the application in better condition for appeal. The incorporated paragraph states that "The invention can also be embodied as computer readable code on a computer readable medium."

In view of the foregoing, the Applicants submit that the term "computer readable media" as used in claims 8-20 is supported by the specification as originally filed. Therefore, the Board of Appeals and Interferences is respectfully requested to overturn the Examiner's rejections of claims 8-20 under 35 U.S.C. 112.

B. Rejections of Claims 1-20 under 35 U.S.C. 101

The Examiner has incorrectly asserted that the method of claim 1 can be performed by a person as a mental step or using pencil and paper. The method of claim 1 recites "A method for automated acquisition of assertions in a specification of a computer program." The term "automated acquisition" as used in claim 1 indicates that the method is not performed manually, i.e., by a person. Thus, the method is not performed by a person as a mental step or using pencil and paper.

Additionally, the Examiner has incorrectly asserted that it cannot be determined from the claim language that any step of the method of claim 1 requires the presence of hardware/machine. The term "automated acquisition" excludes "manual acquisition." For "automated acquisition" to occur, the method operations are by definition automated. Because the method operations are automated, it can be determined from the claim language that the method operations are to be performed automatically by a capable non-human device.

The Examiner has also incorrectly asserted that the language of claim 1 raises a question as to whether the claim is directed merely to an abstract idea that is not tied to a technological art, environment, or machine which would result in a practical application producing a concrete, useful, and tangible result. Claim 1 recites that the sentences marked as testable are used to determine the adequacy of a test suite for testing the computer program. Therefore, claim 1 defines a method having a practical application that produces

a concrete, useful, and tangible result. A claim is limited to a practical application when the method, as claimed, produces a concrete, tangible and useful result, i.e., the method recites a step or act of producing something that is concrete, tangible and useful. *AT&T*, 172 F.3d, 1358, 50 USPQ2d, 1452.

With regard to claims 8 and 14, the Examiner has incorrectly asserted that the claims merely recite a computer program listing corresponding to non-functional descriptive material that is not representative of either a physical thing or a statutory process in which acts are performed. Claim 8 recites a "computer readable media including program instructions for automatically obtaining assertions from a specification of a computer program." Claim 14 recites a "a computer readable media including program instructions for automated acquisition of assertions in a specification of a computer program."

According to MPEP 2106 (IV)(B)(1), functional descriptive material includes data structures and computer programs which impart functionality when employed as a computer component. Non-functional descriptive material includes music, literary works and a compilation or mere arrangement of data. Therefore, the Examiner is first respectfully incorrect in asserting that a computer program is non-functional descriptive material. Rather, a computer program is considered functional descriptive material. Second, the Examiner is respectfully incorrect in asserting that the computer readable media of claims 8 and 14 is non-statutory. According to MPEP 2106 (IV)(B)(1)(a), a claimed computer-readable medium encoded with a data structure defines structural and functional interrelationships between the data structure and the computer software and hardware components which permit the data structure's functionality to be realized, and is thus statutory.

In the basis of rejection, the Examiner stated that the rejected claims were viewed as not meeting a use of technology requirement, in that the claimed steps could be done by pencil and paper. Thus, the rejections were predicated on the existence of a judicially recognized separate "technological arts" test to determine patent eligible subject matter under Section 101. The separate "technological arts" test has been considered in the recently published case of Ex parte Carl A. Lundgren, decided by the Board of Patent Appeals and Interferences based on a hearing on April 20, 2004, and identified by Appeal No. 2003-2088 (the Lundgren case), which is a precedential case.

Referring to the Lundgren case, the Board squarely addressed this issue, as it stated:

> "However, the examiner is of the opinion that there is a separate test for determining whether claims are directed to statutory subject matter, i.e., a "technological arts" test.

The Board reviewed cases cited here by the Examiner. On page 6, the BPAI concluded that:

> We have reviewed these three cases and do not find that they support the examiner's separate "technological arts" test.

Significantly, on page 7, the BPAI held that:

> Our determination is that there is currently no judicially recognized separate "technological arts" test to determine patent eligible subject matter under Section 101. We decline to create one.

Applied to the present facts, because the Lundgren case is a precedential opinion, there is no "technological arts" test. Thus, the Examiner's assertion that the rejected claims do not meet a use of technology requirement, in that the claimed steps could be done by pencil and paper is improper.

In view of the foregoing, the Applicants submit that independent claims 1, 8, and 14 recite statutory subject matter. Accordingly, each of dependent claims 2-7, 9-13, and

15-20 also recite statutory subject matter. Therefore, the Board of Appeals and Interferences is respectfully requested to overturn the Examiner's rejections of claims 1-20 under 35 U.S.C. 101.

C. Rejections of Claims 1-20 under 35 U.S.C. 103(a)

The Examiner has asserted that the Applicants have inappropriately attacked the cited references individually to show non-obviousness, wherein the rejections are based on the combination of cited references. The Applicants disagree with this assertion by the Examiner. More specifically, the Examiner has cited Pavela as teaching each and every feature of the claims, other than the testable assertion describing behavior of an application programming interface that can be tested. The Examiner has cited the MPCD as teaching an application programming interface. However, it should be appreciated that the MPCD is simply a dictionary that provides a definition for an application programming interface. The MPCD does not provide any teaching that associates an application programming interface with a testable assertion.

Furthermore, the Examiner has not relied on the MPCD to teach any other feature of the claims. Therefore, other than providing a definition of an application programming interface, the Examiner has relied upon Pavela to provide all other teachings in asserting that the claims are prima facie obviousness under 35 U.S.C. 103. Accordingly, the Applicants arguments are primarily directed to the teachings, or absence thereof, within Pavela. Thus, it should be appreciated that the Applicants have not inappropriately attacked the cited references on an individual basis. Rather, the Applicants have addressed the reference teachings as they have been asserted by the Examiner.

Pavela discloses a method for generating test cases using a test object library, wherein the test cases are used for testing software. Pavela teaches definition of a source

file that includes a number of tags. Each tag is associated with a particular executable code object that defines a set of instructions for performing a particular test procedure on a particular software program. The source file is used to generate a test plan in a conversational language. The source file is also used to generate an automated test code for testing the particular software program. More specifically, the automated test code is generated using a technique in which commands to system elements are issued and messages responsive to the commands are intercepted and used to provide test results.

It should be appreciated that Pavela is not concerned with and does not teach automated acquisition of assertions in a specification of a computer program, as claimed by the present invention. Rather Pavela is concerned with teaching a method for testing computer software by providing a tool that can be used to develop test cases for the particular computer software to be tested. The combination of Pavela and MPCD fails to render the claims prima facie obvious, as required to support rejection under 35 U.S.C. 103, for at least the reasons discussed below.

With respect to independent claims 1, 8, and 14, Pavela does not teach receiving a specification of a computer program as an input. The Examiner has asserted that the source file of Pavela teaches the specification of a computer program, as required by the present invention. However, the source file of Pavela is created by a user as an input to generate a test plan for software testing and to generate an automated test code. The source file of Pavela is not equivalent to a specification of a computer program. Furthermore, the source file of Pavela is not described as including a plurality of sentences describing a computer program. It should be further noted that a tag or test code object identifier within the source file of Pavela is not equivalent to a sentence or a context, as required by the present invention.

With regard to claim 1, an operation is recited for "receiving the specification as an input, wherein the specification includes a plurality of sentences describing the computer program." Those skilled in the art will appreciate that a specification of a computer program is a conversant textual description of the computer program's functionality. Furthermore, when considering claim 1 in view of the specification, particularly Table 1 and its associated description, it is quite clear that the specification of the computer program refers to a textual description of the computer program's functionality. Furthermore, claim 1 recites that the specification includes a plurality of sentences, i.e., conversant text, describing the computer program.

The Applicants submit that the Examiner has not considered the recited specification of the computer program in view of the specification. During patent examination, the pending claims must be given their broadest reasonable interpretation consistent with the specification. *In re Hyatt*, 211 F.3d 1367, 1372, 54 USPQ2d 1664, 1667 (Fed. Cir. 2000). "Claims are not to be read in a vacuum, and limitations therein are to be interpreted in light of the specification in giving them their 'broadest reasonable interpretation'." 710 F.2d at 802, 218 USPQ at 292 (quoting *In re Okuzawa*, 537 F.2d 545, 548, 190 USPQ 464, 466 (CCPA 1976)) (emphasis in original).

The Examiner has attempted to draw an equivalence between the specification of the computer program as recited in claim 1 and source file (318) as taught by Pavela. The Examiner has asserted that Pavela discloses the source file (318) as being equal to a specification of a computer program at column 2, lines 12-17. However, Pavela (column 2, lines 12-17) states the following:

> "The method comprises the steps of defining a source file having a plurality of tags associated with a member of a library of executable code objects defining a set of instructions for performing a portion of the automatic test procedure, generating a

test plan in a conventional language from the source file, and generating an automated test code for the automated test procedure from the source file."

The above-referenced portion of Pavela does not disclose the source file (318) as being equal to a specification of a computer program. Examination of Pavela's teachings beyond the "Summary of Invention Section" discloses that the source file is in fact a user-generated input file that specifies (via a tag nomenclature) a number of information items associated with generation of a test plan and generation of automated test code that complies with the generated test plan. More specifically, column 5 of Pavela describes how the source file is generated manually by a user to include information such as test case objectives, scenario, procedure, system configuration, and the parts used by the test case. Figure 4 of Pavela actually shows an example template of the source file. Furthermore, Pavela (column 5, lines 8-9 and 14-15) teaches that the source file is used as an input to generate a test plan and automated test code associated with the test plan. Therefore, in contrast to the present invention, the source file description as provided by Pavela does not indicate that the source file represents a specification of a computer program, wherein the specification includes a plurality of sentences describing the computer program. Rather, the source file of Pavela is disclosed as simply storing input for generating a test plan and associated test code.

Further with respect to claims 1, 8, and 14, Pavela does not teach determining whether a sentence obtained from the specification of the computer program is a testable assertion, wherein the testable assertion describes behavior of an application programming interface that can be tested. However, the Examiner has asserted that Pavela (column 2, lines 12-17, and column 6, lines 28-30) in combination with MPCD teaches the above-referenced claim feature.

The Examiner has asserted that the tag as disclosed by Pavela is equivalent to a testable assertion. However, the referenced teachings of Pavela merely indicate that the tags in the source file of Pavela can be associated with members of a library of executable test code objects. However, the tag of Pavela is described as an entity used to identify and delineate information entered in the source file. The tag of Pavela does not describe behavior of a testable application programming interface, as required by the testable assertion of the present invention. The Examiner has simply cited a definition of application programming interface in the MPCD reference in an attempt to convey a particular meaning to the tag as disclosed by Pavela, such that the tag of Pavela may be misconstrued as teaching the testable assertion recited by the present invention. However, simply citing a definition of application programming interface does not infer an meaning to the term testable assertion, which is itself absent from the teachings of Pavela.

Furthermore, there must be some suggestion or motivation within the cited references themselves to combine their respective teachings in establishing a case of prima facie obviousness against a claim. Because the MPCD is simply a dictionary that provides definitions for various terms, it should be appreciated that whatever motivation exists for combining the teachings of Pavela with the MPCD must derive from Pavela. However, Pavela is silent with regard to the application programming interface, particular as it relates to the tags within the source file as taught by Pavela. Therefore, the Applicants submit that there is no motivation to combine the definition of application programming interface as provided by the MPCD reference with the teachings of Pavela in asserting that the tag of Pavela teaches the testable assertion recited by the present invention.

There must be some suggestion either explicitly or implicitly in the references themselves to combine their teachings as claimed. The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior

art also suggests the desirability of the combination. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990).

Additionally, it should be appreciated that the tags in the source file of Pavela do not represent a sentence describing a computer program. Therefore, it is not reasonable to consider the tags of Pavela as representing a testable assertion, wherein the testable assertion is one of a plurality of sentences describing the behavior of an application programming interface that can be tested, as required by the presently claimed invention. Furthermore, the referenced teachings of Pavela provide no mention of performing a determination as to whether a sentence represents a testable assertion.

Furthermore, the Examiner has asserted that generation of the test index as taught by Pavela is equivalent to the feature of claim 1 for "marking the obtained sentence as testable when the obtained sentence is a testable assertion." Because Pavela does not teach obtaining sentences from the specification of a computer program and determining whether the obtained sentences represent a testable assertion, it is not reasonable to conclude that Pavela teaches marking the obtained sentences as testable when the obtained sentence represents a testable assertion. The test index generated in Pavela simply represents a listing of system elements that are tested by a test case, wherein the test case is defined manually using the source file as an input vehicle.

Further with respect to claim 8, Pavela does not teach identification of a context within the specification of the computer program. Additionally, Pavela does not disclose parsing the context within the specification of the computer program to obtain sentences.

To establish prima facie obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). Based at least on the above arguments, the Applicants submit that the combination of Pavela and MPCD fails to teach each and every feature of claim 1,

as required to render claim 1 prima facie obvious. Furthermore, the Applicants submit that the above arguments provided for claim 1 are equally applicable to similar features recited in each of claims 8 and 14. Therefore, based at least on the above arguments, the Applicants submit that the combination of Pavela and MPCD fails to teach each and every feature of claims 8 and 14, as required to render claims 8 and 14 prima facie obvious. Additionally, the Applicants submit that each of dependent claims 2-7, 9-13, and 15-20 is patentable for at least the reasons provided for its respective independent claim. The Board of Appeals and Interferences is respectfully requested to overturn the Examiner's rejections of claims 1-20 under 35 U.S.C. 103.

In view of the foregoing, the Applicants submit that each of claims 1-20 is patentable. Therefore, the Applicants respectfully request that the Board of Appeals and Interferences reverse the Examiner's rejections of the claims on appeal.

Respectfully Submitted,
MARTINE PENILLA & GENCARELLA, LLP

Kenneth D. Wright
Reg. No. 53,795

MARTINE PENILLA & GENCARELLA, LLP
710 Lakeway Drive, Suite 200
Sunnyvale, California 94085
408.749.6900

VIII. CLAIMS APPENDIX

1. A method for automated acquisition of assertions in a specification of a computer program, comprising:

receiving the specification as an input, wherein the specification includes a plurality of sentences describing the computer program;

obtaining a sentence from the plurality of sentences;

determining whether the obtained sentence is a testable assertion, wherein the testable assertion describes behavior of an application programming interface that can be tested;

marking the obtained sentence as testable when the obtained sentence is a testable assertion; and

using the sentences marked as testable to determine whether a test suite for testing the computer program is adequate.

2. The method as recited in claim 1, further comprising:

identifying a context within the specification.

3. The method as recited in claim 2, wherein the operation of obtaining the sentence from the plurality of sentences includes parsing the context to obtain the sentence.

4. The method as recited in claim 3, further comprising:

adding the marked obtained sentence to an assertion result set.

5. The method as recited in claim 4, wherein the context is a set of circumstances related to the obtained sentence.

6. The method as recited in claim 5, wherein each assertion includes at least one sentence of the specification.

7. The method as recited in claim 5, wherein each assertion includes at least two sentences of the specification.

8. A computer readable media including program instructions for automatically obtaining assertions from a specification of a computer program, comprising:

a code segment that receives the specification as an input;

a code segment that identifies a context within the specification;

a code segment that parses the identified context to obtain sentences;

a code segment that determines whether the obtained sentences are testable assertions, wherein each testable assertion is a sentence that describes behavior of an application programming interface that can be tested; and

a code segment that adds the testable assertions to an assertion result set, wherein the assertion result set can be used to facilitate testing of the specification.

9. The computer readable media of claim 8, further comprising:

a code segment that filters the identified context prior to parsing the context.

10. The computer readable media of claim 9, wherein the code segment that receives the specification is defined to receive the specification in a text format.

11. The computer readable media of claim 9, wherein the context is a set of circumstances related to the obtained sentences.

12. The computer readable media of claim 9, wherein each assertion includes at least one sentence of the specification.

13. The computer readable media of claim 9, wherein each assertion includes at least two sentences of the specification.

14. A computer readable media including program instructions for automated acquisition of assertions in a specification of a computer program, comprising:

a code segment that receives the specification in a text format, wherein the specification includes a plurality of sentences;

a code segment that obtains a sentence from the plurality of sentences;

a code segment that determines whether the obtained sentence is a testable assertion, wherein the testable assertion describes behavior of an application programming interface that can be tested; and

a code segment that marks the obtained sentence as testable when the obtained sentence is a testable assertion.

15. The computer readable media of claim 14, further comprising:

a code segment that identifies a context within the specification.

16. The computer readable media of claim 15, wherein the code segment that obtains the sentence from the plurality of sentences includes a code segment that parses the context to obtain the sentence.

17. The computer readable media of claim 16, further comprising:

a code segment that adds the marked obtained sentence to an assertion result set.

18. The computer readable media of claim 17, wherein the context is a set of circumstances related to the obtained sentence.

19. The computer readable media of claim 18, wherein each assertion includes at least one sentence of the specification.

20. The computer readable media of claim 19, wherein each assertion includes at least two sentences of the specification.

IX. EVIDENCE APPENDIX

There is currently no evidence entered and relied upon in this Appeal.

X. RELATED PROCEEDINGS APPENDIX

There are currently no decisions rendered by a court or the Board in any proceeding identified in the Related Appeals and Interferences section.